

普通高校计算机主干课程辅导与提高丛书

任爱华 编著

操作系统 辅导与提高



清华大学出版社

普通高校计算机主干课程辅导与提高丛书

操作系统辅导与提高

任爱华 编著

清华大学出版社

北 京

内 容 简 介

为了帮助学生对操作系统原理的学习和理解,本书以不同的专题讲解在学习过程中遇到的难题,通过对这些问题的思考和解决,使学生能充分地理解和掌握操作系统的基本原理,以及解决操作系统问题的基本原则和方法,增强分析问题与解决问题的能力。

全书共分8个专题和1个附录。内容包括操作系统的总体框架、引导程序的设计和进程环境的建立实例、进程模型的建立和并发程序设计、有关存储管理的重要概念及其管理的实例、设备管理应用方面的例子以及设备驱动程序编写、文件管理方面的应用问题及其编程实例、操作系统用户接口方面的相关实例以及添加Linux的系统调用的方法,还包括有关实时系统、机群系统以及Agent系统方面的应用编程实例,对这些系统的应用有助于学生从更高层次上理解操作系统的基本原理和应用。

附录中提供了操作系统综合练习题和参考答案,另外还有实例中用到的完整源程序以及相关的自由软件(请在 www.thjd.com.cn 的“源代码下载”处下载)。

本书适合高等院校计算机专业的学生及参加计算机研究生入学考试的学生参考。

版权所有,翻印必究。

本书封面贴有清华大学出版社激光防伪标签,无标签者不得销售。

图书在版编目(CIP)数据

操作系统辅导与提高/任爱华编著. —北京:清华大学出版社,2004

(普通高校计算机主干课程辅导与提高丛书)

ISBN 7-302-08103-4

I. 操… II. 任… III. 操作系统—高等学校—教学参考资料 IV. TP316

中国版本图书馆CIP数据核字(2004)第011174号

出 版 者:清华大学出版社 地 址:北京清华大学学研大厦

<http://www.tup.com.cn> 邮 编:100084

社 总 机:010-62770175 客 户 服 务:010-62776969

责任编辑:刘利民

封面设计:钱 诚

版式设计:杨 洋

印 刷 者:北京市昌平环球印刷厂

装 订 者:三河市昭亮装订有限公司

发 行 者:新华书店总店北京发行所

开 本:185×260 印张:22.25 字数:492千字

版 次:2004年3月第1版 2004年3月第1次印刷

书 号:ISBN 7-302-08103-4/TP·5858

印 数:1~5000

定 价:28.00元

本书如存在文字不清、漏印以及缺页、倒页、脱页等印装质量问题,请与清华大学出版社出版部联系调换。联系电话:(010)62770175-3103 或 (010)62795704

普通高校计算机主干课程辅导与提高丛书

组织与编著委员会

主 编 徐孝凯

成 员 (排列无先后)

任爱华 (北京航空航天大学)

宿红毅 (北京理工大学)

盛定宇 (首都经贸大学)

顾一禾 (南京理工大学)

王小铭 (华南师范大学)

刘世峰 (北京交通大学)

何 军 (中国人民大学)

郑 岩 (北京邮电大学)

顾 问 许卓群 (北京大学)

侯炳辉 (清华大学)

丛 书 序

随着我国国民经济持续、稳定、快速地发展,我国高等教育正在经历一个前所未有的发展阶段,招生规模不断扩大。在世界范围内,信息技术得到了空前发展和广泛应用,社会迫切需要培养大量的计算机和信息技术方面的人才。计算机和信息类专业已经成为我国当前高等教育中社会最需要、发展最迅速、招生规模最大的热门专业之一。

由于计算机和信息类专业招生规模扩大,高校师资力量显得不足,讲授课程只能由小班改为大班或合班上课,对每个学生的批改作业时间和辅导答疑时间也相应地减少,但学生所使用的教材没有适应这种变化,以前学习计算机专业教材就比较难,现在相对难度更大了。这就要求配合使用针对性强的、便于自学的辅导教材,来帮助解决在学习过程中遇到的各类问题,提高学习效率。现在社会上计算机类的教材比较多,但用于学生自主学习的辅助教材却很少,各地师生要求尽快组织出版此类教材的呼声越来越高。清华大学出版社根据学生的迫切需要,本着对学生高度负责的精神,组织一批具有相关课程教学经验和编写教材经验并深受读者好评的高校专家编写了这套辅助教材。

这套书将分批陆续推出,首批推出6本。它们是《计算机组成原理辅导与提高》、《数据结构辅导与提高》、《操作系统辅导与提高》、《计算机网络辅导与提高》、《数据库原理辅导与提高》、《软件工程辅导与提高》。

这套书的编委会从有利于学生自主学习的需要出发,经过多次酝酿和研讨,最后形成统一认识,采用统一的编写格式,主要特点如下:

- ◆ 从每门课程的学科内容出发,针对一般教材中的重点和难点,分专题进行循序渐进地解剖和分析,并尽量通过恰如其分的事例进行说明,使读者既能够理解和掌握重要的概念,又能够从理论和实际应用的结合上加深认识。
- ◆ 为了加强训练和应用,巩固所学的知识,在每本教材的附录中分别给出了针对课程内容的练习题和参考答案。练习题型丰富,包含选择、填空、运算、分析、设计等题型,对于较难的习题在解答中同时给出了分析过程。
- ◆ 为了自测本人的学习效果,在附录中给出了一套试卷,要求2小时内做完。
- ◆ 本套丛书的读者对象为学习相应课程的本、专科学生以及考研的学生,他们可以根据各自需要选取有关内容。

欢迎广大读者对本丛书提出宝贵意见,我的电子邮件地址为 xuxk@crtvu.edu.cn。

丛书主编 徐孝凯

2003年12月

前 言

操作系统在用户和计算机硬件之间建立了桥梁，用户通过操作系统命令交互地使用计算机，也通过操作系统的编程界面——系统调用使用操作系统内部提供的功能程序。在学习和掌握操作系统设计原理的过程中，有许多抽象的概念难以理解和实践。对操作系统的学习和实践需要学生具有使用计算机的经历和基本的编程技能，需要了解计算机的组成以及计算机硬件提供了哪些编程使用的接口，这些接口为第一层软件——操作系统的开发提供了程序控制基本支持。为了帮助学生对操作系统原理的学习和理解，本书以不同的专题讲解在学习过程中遇到的难题，通过对这些问题的思考 and 解决，使学生能充分地理解和掌握操作系统的基本原理，以及解决操作系统问题的基本原则和方法，增强分析问题与解决问题的能力。

全书共分 8 个专题和 1 个附录。专题 1 是操作系统的总体框架；专题 2 是引导程序的设计和进程环境的建立实例；专题 3 是进程模型的建立和并发程序设计；专题 4 介绍一些有关存储管理的重要概念和有关屏蔽硬件的数据结构，最后给出一个利用系统调用申请内存，并对该内存区域进行管理的实例；专题 5 首先介绍一个设备管理应用方面的例子，然后介绍如何编写一个设备驱动程序；专题 6 涉及文件管理方面的应用问题，并提供一个模拟文件系统的编程实例；专题 7 是关于操作系统用户接口方面的相关实例，介绍如何添加 Linux 的系统调用以及如何编写 shell 解释程序；在专题 8 提供了有关实时系统、机群系统以及 Agent 系统方面的应用编程实例，对这些系统的应用有助于学生从更高层次上理解操作系统的基本原理和应用。

附录中提供了操作系统综合练习题和答案，另外还有实例中用到的完整源程序以及相关的自由软件（请在 www.thjd.com.cn 的“源代码下载”处下载）。

为了便于读者理解习题，每一章都以基本概念的简单陈述开始，然后进入实例应用和解答。

在本书中，专题 1、4、7 由任爱华执笔，专题 2 由陈佩、刘辉执笔，在其他专题中的 Linux 实验例子以及 RTLinux 的应用实例由李鹏执笔，Windows 实验例子由孙云峰与刘方毅执笔，模拟文件系统的例子由王菲执笔，Aglets 的应用实例由孙云峰执笔，PVM 的应用实例以及网格应用实例由石宏义执笔。全书由任爱华进行统一修改、审校并统稿。

限于编著者的水平，错误与不妥之处在所难免，恳求读者批评指正。

任爱华

2004 年 1 月于北京

目 录

专题 1 操作系统的总体框架	1
1.1 绪论	1
1.2 计算机系统的层次划分	1
1.3 操作系统提供了抽象的计算环境	3
1.4 操作系统的总体结构	5
1.4.1 硬件编程接口以及操作系统接口	5
1.4.2 操作系统设计层次	7
1.4.3 操作系统基础实现的考虑	9
1.4.4 Linux 操作系统的整体结构	12
1.4.5 Windows 操作系统的整体结构	13
1.5 小结	16
1.6 习题	17
专题 2 引导程序的设计和进程环境的建立实例	18
2.1 引导程序设计原理与实例	19
2.1.1 引导程序实现原理	19
2.1.2 引导程序编写	37
2.1.3 参考表格	50
2.1.4 习题	51
2.2 进程模型的建立	52
2.2.1 实现进程的原理	52
2.2.2 在 80386 上进程的实现例程	61
2.3 小结	73
2.4 习题	74
专题 3 进程模型的建立和并发程序设计	75
3.1 在 Windows 系统下用高级语言创建多进（线）程	75
3.1.1 在 Java 中创建线程的方法	76
3.1.2 在 .NET 平台上创建线程的方法	77
3.1.3 Delphi 创建线程的方法	79
3.1.4 VC 中的线程创建	81
3.1.5 VC 中的进程创建	86

3.2 几个经典的同步互斥问题.....	92
3.2.1 基础知识.....	93
3.2.2 线程间通信的实现.....	95
3.3 在 Linux 系统下实现进程间通信.....	107
3.3.1 管道.....	107
3.3.2 FIFO.....	111
3.3.3 消息队列.....	114
3.3.4 信号量.....	120
3.3.5 共享存储.....	129
3.4 小结.....	134
3.5 习题.....	135
专题 4 存储管理应用实例.....	136
4.1 概述.....	136
4.2 存储管理对内存硬件的抽象.....	137
4.3 用户编程中申请与释放内存实例分析.....	140
4.3.1 Malloc.h 文件.....	140
4.3.2 Malloc.c 文件.....	141
4.3.3 Test.c 文件.....	145
4.3.4 Makefile 文件.....	145
4.4 小结.....	146
4.5 习题.....	146
专题 5 设备管理.....	147
5.1 设备管理的基本概念.....	147
5.2 使用 Windows 系统中的磁盘输入/输出.....	150
5.2.1 实验要求.....	150
5.2.2 设计方案介绍.....	150
5.2.3 程序编译及运行.....	151
5.2.4 运行结果解释.....	152
5.2.5 程序用到的系统 API 函数.....	154
5.2.6 源代码示例.....	157
5.3 怎样编写 Linux 下的设备驱动程序.....	162
5.3.1 设备驱动程序的接口.....	163
5.3.2 设备驱动程序的编译和加载.....	164
5.3.3 实例介绍.....	165
5.4 小结.....	181

5.5 习题	181
专题 6 文件管理	182
6.1 文件系统综述	182
6.2 模拟文件管理系统实例	186
6.2.1 实验要求	186
6.2.2 设计方案	187
6.2.3 编译及运行	193
6.2.4 源代码	193
6.3 小结	212
6.4 习题	212
专题 7 操作系统接口	213
7.1 操作系统接口	213
7.1.1 系统调用	213
7.1.2 Shell 命令及其解释程序	222
7.2 Linux 的安装	232
7.2.1 安装前的准备	233
7.2.2 建立硬盘分区	233
7.2.3 安装类型	235
7.2.4 安装过程	235
7.2.5 操作系统的安装概念	236
7.3 Linux 的使用	236
7.3.1 使用常识	236
7.3.2 文件操作命令	237
7.3.3 文本编辑命令	243
7.3.4 Shell 的特殊字符	246
7.3.5 进程控制命令	250
7.3.6 网络应用工具	252
7.3.7 联机帮助	254
7.4 系统管理	254
7.4.1 超级用户	255
7.4.2 用户和用户组管理	255
7.4.3 文件系统管理	258
7.4.4 Linux 源代码文件安放结构	261
7.5 小结	262
7.6 习题	262

专题 8 实时系统、集群、网格、Agent 简介与使用	263
8.1 实时操作系统 RTLinux	263
8.1.1 简介	263
8.1.2 RTLinux 安装	267
8.1.3 编写 RTLinux 程序	269
8.2 集群及 PVM 介绍	271
8.2.1 集群的概念	271
8.2.2 PVM 的产生和发展	272
8.2.3 PVM 的特点	272
8.2.4 PVM 的系统组成	273
8.2.5 PVM 的安装和使用	273
8.2.6 PVM 的实验	275
8.3 网格及 Globus 介绍	287
8.3.1 网格概念	287
8.3.2 网格系统构成	288
8.3.3 Globus	289
8.3.4 实验	294
8.4 移动代理编程简介	300
8.4.1 引言	300
8.4.2 Aglets 平台的安装	300
8.4.3 Aglets 的系统架构	301
8.4.4 Aglets 的对象模型	302
8.4.5 Aglets 的常用函数	303
8.4.6 试验	303
8.5 小结	305
8.6 习题	305
操作系统练习题	306

专题 1 操作系统的总体框架

1.1 绪 论

从取缔操作员的手工操作开始，到目前的多道程序设计系统（并发系统），这一过程形成了操作系统的发展史。

从操作系统提供给用户的程序接口上看，任何操作系统的核心内容都体现在一组系统调用上，系统调用表达了操作系统真正执行的内容。程序员通过系统调用使用操作系统程序，系统调用一般划分为进程管理（如建立进程与终止进程）、存储管理（如申请内存）、文件管理（如文件的建立、读、写、删除等）和设备管理（如安装设备驱动程序等）几个部分。

从操作系统本身的设计角度来看，操作系统有若干种构造模式，常见的有整体式内核（Monolithic kernel）模式和微内核（Micro-kernel）模式，整体式内核也称为宏内核或者单内核。

整体式内核在运行过程中是一个单独的内存映像，也是一个独立的进程。Linux 系统属于这种整体式内核构造模式，Linux 采用的是模块接口法设计技术。

微内核在运行过程中，大部分内核模块都作为独立的进程，它们之间通过消息通信在模块之间提供服务。微内核本身类似一个消息管理器，设计上具有灵活性，易于内核模块的维护和移植；其缺陷则是消息传递开销引起效率的降低。Windows 操作系统就是微内核构造模式，采用的是面向对象设计技术。

1.2 计算机系统的层次划分

计算机系统包括计算机硬件和软件。用户通过软件层提供的界面告诉计算机为用户做些什么工作，也就是通过操作系统来与计算机交互用户的意图。计算机的硬件和软件由操作系统进行统一管理。一个典型的 PC 机硬件结构如图 1.1 所示，它包括了 CPU、内存、输入/输出设备以及连接这些部件的总线。在硬件层之上是软件层，或称为资源抽象层，包括操作系统软件和系统软件，用户编写的应用软件依赖于资源抽象层，如图 1.2 所示。

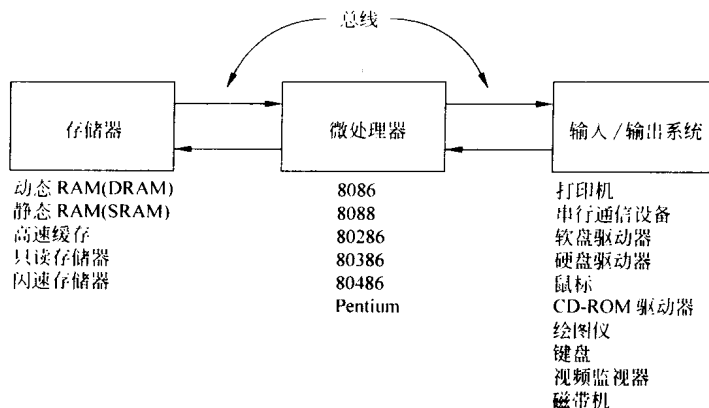


图 1.1 典型的 PC 机硬件结构

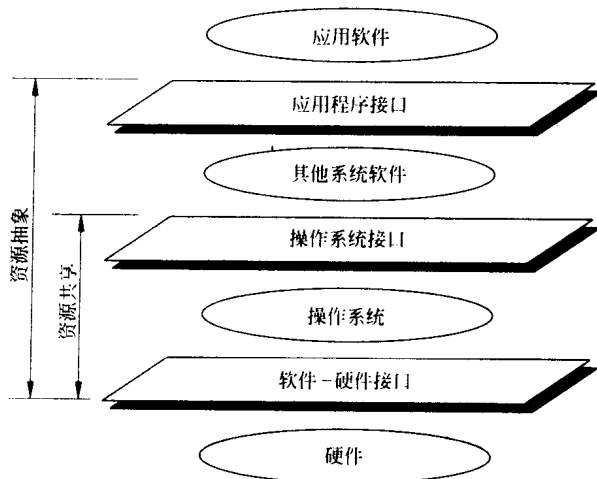


图 1.2 系统软件和操作系统软件

计算机硬件为用户提供了操作的物理基础，程序员通过硬件编程接口来控制硬件，这些硬件编程接口是指：机器指令集合、各种寄存器、内存地址以及设备地址等，这些地址接口分别与存储器和设备控制器相连，而设备控制器又通过硬件端口与设备相连；操作系统在硬件之上扩充了硬件编程接口能够为用户提供的功能，操作系统对计算机硬件资源进行了抽象或者透明化，如图 1.3 所示。所以，用户可以通过操作系统接口，即系统调用和交互式命令来方便地使用计算机的硬件，而无需了解对硬件控制的具体细节，但是操作系统的设计人员则需要了解这些硬件编程接口的使用方法。系统软件在操作系统的基础上又增加了新的功能，通过应用程序接口提交给用户使用，不同的系统软件在操作系统基础之上进一步抽象操作系统软件资源，为用户提供更加集成的功能和更加方便地使用计算机的手段。当然操作系统本身也是系统软件，不过这里提到的系统软件是指除了操作系统以外的系统软件，如编译程序、汇编程序、调试程序、编辑程序等，它们也可称为实用程序。

像微软 VC++ 这样的工具软件是这类实用程序的集成环境。

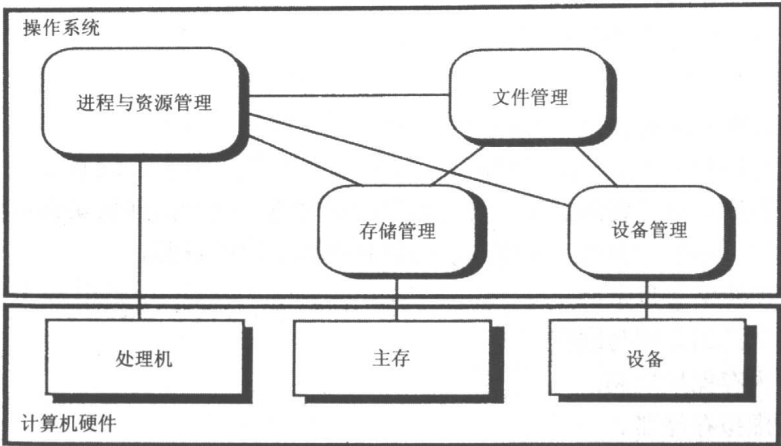


图 1.3 操作系统是对硬件资源的抽象

1.3 操作系统提供了抽象的计算环境

操作系统既是对硬件的抽象，也是对计算机软件和硬件进行管理的程序，所以操作系统由对硬件资源的管理模块和软件资源的管理模块构成，这些模块分别是进程管理、存储管理、设备管理和文件管理，这些模块之间的关系以及它们与硬件的抽象关系如图 1.4 所示。

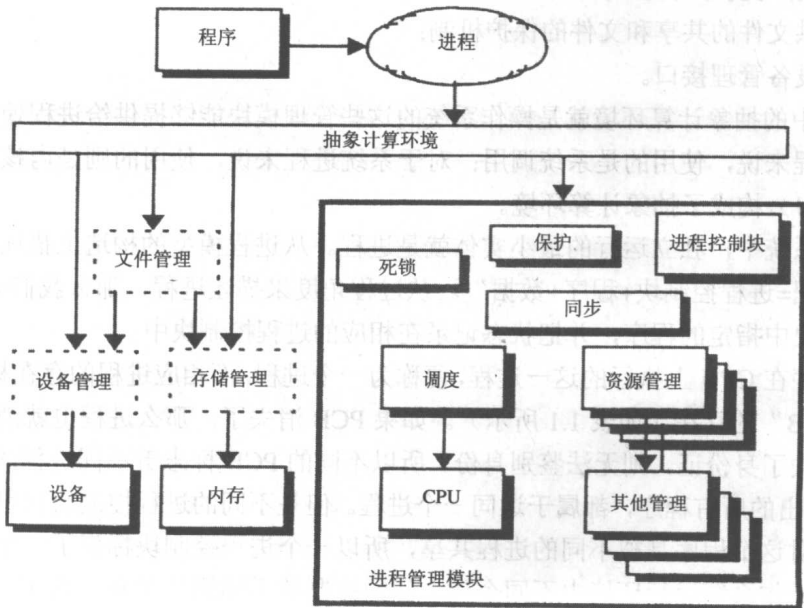


图 1.4 进程管理控制下的软件和硬件之间的关系

在图 1.4 中，进程管理模块主要负责：

- 提供基本算法和数据结构，实现进程和资源的抽象；
- 分配进程使用处理机的时间，即进程调度；
- 协调进程之间的相互制约关系，即进程之间的同步与互斥的控制；
- 检测进程之间是否有永久的相互等待出现，即死锁检测；
- 负责对各种资源的分配与释放。有多少不同种类的资源，就有多少个不同的资源管理程序，这里资源是指：凡是有可能阻塞进程执行的任何实体都可称为资源，像磁盘、磁带称为可重用资源，而消息则为消耗性资源。

存储管理模块主要负责：

- 内存资源的分配与回收；
- 提供内存保护机制；
- 实现虚拟存储器。

设备管理模块主要负责：

- 数据传输控制；
- 中断处理；
- 设备驱动程序；
- 缓冲区管理。

文件管理模块主要负责：

- 从文件逻辑结构到文件物理组织的映射；
- 目录管理；
- 外存分配与释放；
- 为用户提供文件操作命令；
- 提供文件的共享和文件的保护机制；
- 与设备管理接口。

图 1.4 中的抽象计算环境就是操作系统的这些管理模块能够提供给进程使用的接口，对于用户进程来说，使用的是系统调用；对于系统进程来说，使用的则是内核调用。这些调用（即接口）构成了抽象计算环境。

在操作系统中，独立运行的最小实体就是进程。从进程模型的构造上描述进程，可以描述为“进程=进程控制块+程序+数据”。从过程角度来描述进程，那么我们说，CPU 执行进程控制块中指定的程序，并把状态记录在相应的进程控制块中。

这个程序在 CPU 上执行的这一过程，便称为一个进程，而相应进程的存在标志就是“进程控制块 PCB”的存在（如表 1.1 所示）。如果 PCB 消失了，那么进程也就消失了，就如同一个人丢失了身份证，则无法鉴别身份。所以不同的 PCB 标志着不同进程的存在，在一个 PCB 下指出的所有程序，都属于这同一个进程。但是不同的进程控制块也可能指向同一个程序，此时这个程序是被不同的进程共享，所以一个进程控制块标识了一个进程。进程集合构成了操作系统运行环境中不同个体之间的控制关系和裙带关系，操作系统提供的运行环境就如同一个“进程”构成的单一社会，而“进程”实体便是组成这个进程“社会”

的独立成员。所以，当观察运行中的操作系统时，能够看到的、或者能够控制的便是以进程为独立个体的一个运行环境，CPU 执行的每一条指令、每一个程序都是有主的，即一定是归属于某个特定进程。

在整个计算机系统中，操作系统是硬件之上的第一层系统软件，它提供了对硬件资源的抽象，使用户对计算机硬件的使用透明化，方便了用户对计算机的操作过程。因此，通常只要在计算机系统中添加一个新的软件，便认为又得到了一个新的虚拟机，操作系统是紧接在计算机系统的硬件层之上添加的第一个软件，所以它与原有计算机硬件系统一起形成了第一个虚拟机。

表 1.1 Linux 中的进程控制块 (task_struct 结构)

进程标识
进程状态
调度信息
进程链
时间片
进程通信信息
内存资源信息
文件资源信息
进程上下文

1.4 操作系统的总体结构

1.4.1 硬件编程接口以及操作系统接口

操作系统是为了方便用户使用计算机以及为了使计算机资源得到有效和充分的利用而设计的，所以操作系统的设计依赖于硬件提供的环境以及取决于用户对计算机的使用所提出的目标。

从能够提供的硬件环境到用户使用计算机所要达到的目标两方面来看，操作系统的设计涉及两个层面上的接口问题，即它自身能够提供的接口以及它所依赖的硬件接口。

硬件层提供的编程接口：这是操作系统设计人员需要掌握的硬件基础，操作系统软件通过控制这些编程接口达到控制硬件的目的。

操作系统软件为用户提供使用的接口：这是计算机的用户需要掌握的内容，通过操作系统接口（系统调用或实用程序），用户可以利用操作系统来控制计算机。

我们可以从整个计算机系统的不同层次的视图，进一步认识操作系统程序设计人员与普通用户的程序员之间需要了解的接口知识上的差别，如图 1.5 所示。在硬件层上，操作

系统设计者需要了解的基础内容如图 1.6 所示。

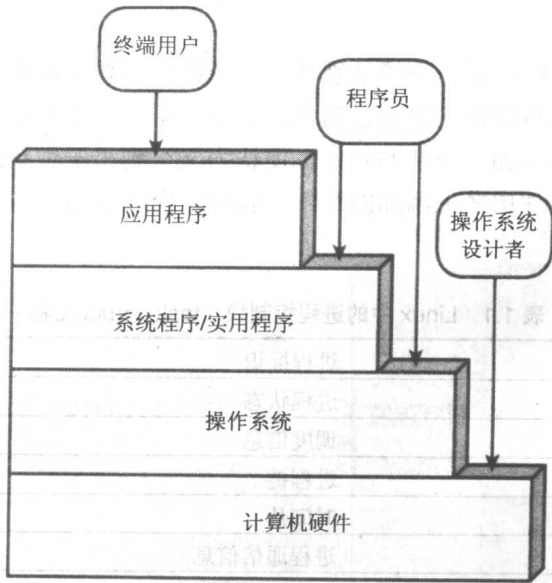


图 1.5 计算机系统的层次和视图

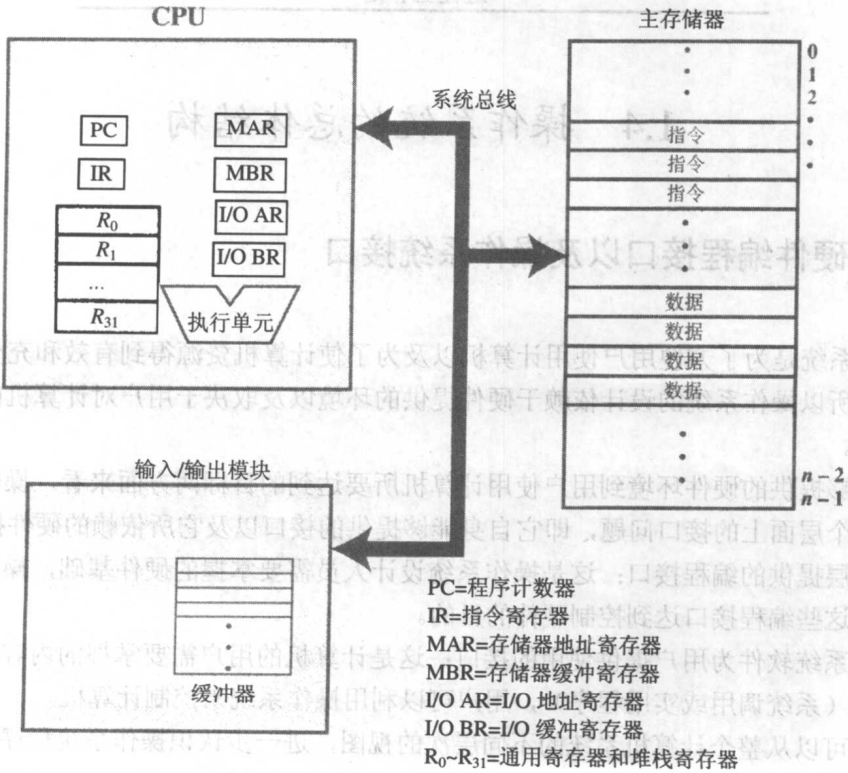


图 1.6 计算机部件程序接口

从图 1.6 中可以看到, 硬件部分如果是可以进行程控的, 则必须提供寄存器和指令系统, 所以各种各样的寄存器和指令系统的使用方法是操作系统设计人员需要了解的硬件基础, 这些寄存器便是操作系统通过指令控制的硬件接口, 而硬件部分通过这些寄存器得到控制信号, 从而进行动作。

另一方面, 用户所使用的操作系统接口, 则是操作系统自身完成的功能对用户开放的部分, 开放的形式是以系统调用的手段提供给用户使用。提供的功能范围包括了进程控制、文件操作、存储分配等功能。这些系统调用呈现的形式依赖于汇编语言中的中断指令, 如果提供更加方便一些的调用形式, 则可体现在各种高级语言之中, 比如, Linux 中的系统调用可以用 C 函数的形式提供给用户使用, 在 Windows 中操作系统调用可以在各种高级语言中以 API (应用程序接口) 的形式出现, 对于这些形式的操作系统调用可参见专题 3 中的各种应用实例。

1.4.2 操作系统设计层次

随着底层硬件朝着更加通用和功能更加强大的趋势发展, 操作系统的规模和复杂性也会随之增加。经常会出现的问题是: 新的操作系统在交付使用时, 其功能已经呈现出落后; 而且系统还会有潜在的错误; 关于预期的性能也会存在着差异。由于这些因素的存在, 都促使操作系统的设计走向更加模块化、层次化以及对象化。

操作系统的层次结构按照复杂性、时间常数、抽象性将功能分开, 我们可以把操作系统看作是一系列的层, 每一层由一些小的功能模块组成, 执行操作系统所需功能的相关子集。每层都依赖于其下一层, 而较低层执行更为原始的功能并隐藏这些功能的细节, 每层为其上一层提供服务。理论上讲, 通过对层的定义, 可以使得改变一层时无需改变其他层, 这样就把一个问题分解成几个更易于处理的子问题。

总之, 较低层的处理时间越短越好, 将直接与硬件打交道的程序部分放在最底层。

应用这些原理的方式在不同的操作系统中有很大的不同。但是, 为了获得操作系统的一个概貌, 这里给出一个层次操作系统模型供读者参考, 这个模型并不对应特定的操作系统, 只是作为参考。模型的定义如表 1.2 所示, 它由下面几层组成。

表 1.2 操作系统设计层次

层	名 称	对 象	示 例 动 作
13	shell	用户程序设计环境	shell 语言中的语句
12	用户进程	用户进程	退出、终止、挂起和恢复
11	目录	目录	创建、删除、链接、分立、查找和列举
10	设备	外部设备, 如打印机、显示器和键盘	打开、关闭、读和写
9	文件系统	文件	创建、销毁、打开、关闭、读和写
8	通信	管道	创建、销毁、打开、关闭、读和写